vmware EXPLORE

Advanced lab: Using ChatGPT to write Powershell Automation

By Eric Nielsen

vmware EXPLORE

“Join VMware{code} @comminty.broadcom.com”

@ericnipro

- ***University of Florida**; Computer Science & Marketing “85”*
- ***Stanford**: Graduate Classes in Engineering*
- **Out of College**: “IBM ESD, PC/XT/AT” Business Apps Group
- **Companies**: “Gould Electronics, Kodak, Pepsi Co, Sun Microsystems
- **Consulting**: US Navy, Australian Navy, Interactive Systems, others.
- **Founded:** BigAdmin, Flash Archive “flar –cvf”, vExpert, VMware{code}

For fun

- “Bulletin Boards for sharing “ 1 upload for 10 downloads
- Got my first SCO Unix box on a “6MHz IBM AT” first “root” password
- “UUCP to Universtiy of Miami” for my own email feed.
- Started learning electronics and have never looked back
- Married, w/four children, “My oldest is a hardware engineer at Roku”

Part of the “Any Devic

Summary of this session

- Software Languages
- Understanding User Rolls
- Exercising ChatGPT as a Software Dev.

vmware EXPLORE

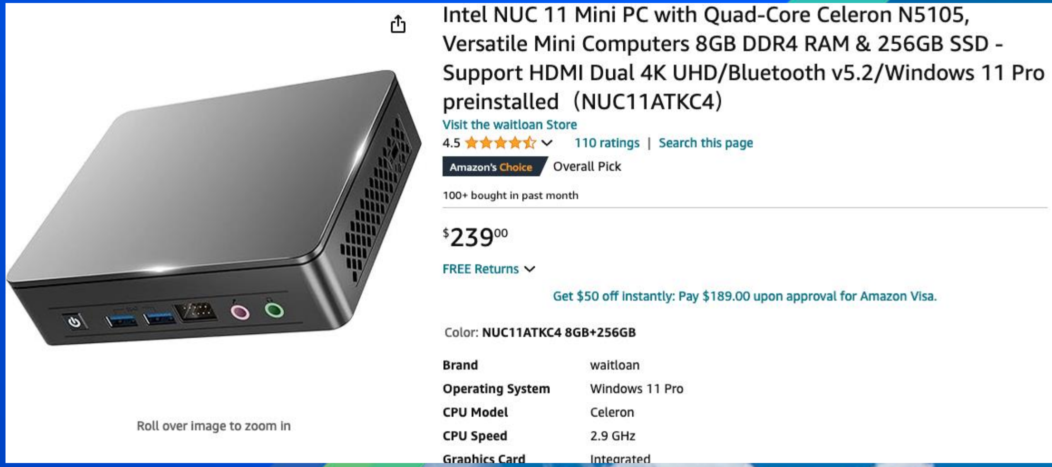
The Hardware for this Lab

- Intel NUC/Clone with Celeron X86 CPU
- 16/32/64 Gig of Memory, No fan, Dual HDMI, Wifi USB Ports
- Can Run ESX with USB Ethernet Dongle
- 256/500 Gig SSD
- 1080p Display (115.00) , Keyboard, Mouse

- Why: <280.00 for a quiet home lab machine

  Portable, one entire table fits in two backpacks (No shipping)
  Community setup, no event labor required.

  More versatile than a Raspberry PI 4/5



Intel NUC 11 Mini PC with Quad-Core Celeron N5105, Versatile Mini Computers 8GB DDR4 RAM & 256GB SSD - Support HDMI Dual 4K UHD/Bluetooth v5.2/Windows 11 Pro preinstalled（NUC11ATKC4）

Visit the waitloan Store

4.5 ★★★★☆ 110 ratings | Search this page

Amazon's Choice Overall Pick

100+ bought in past month

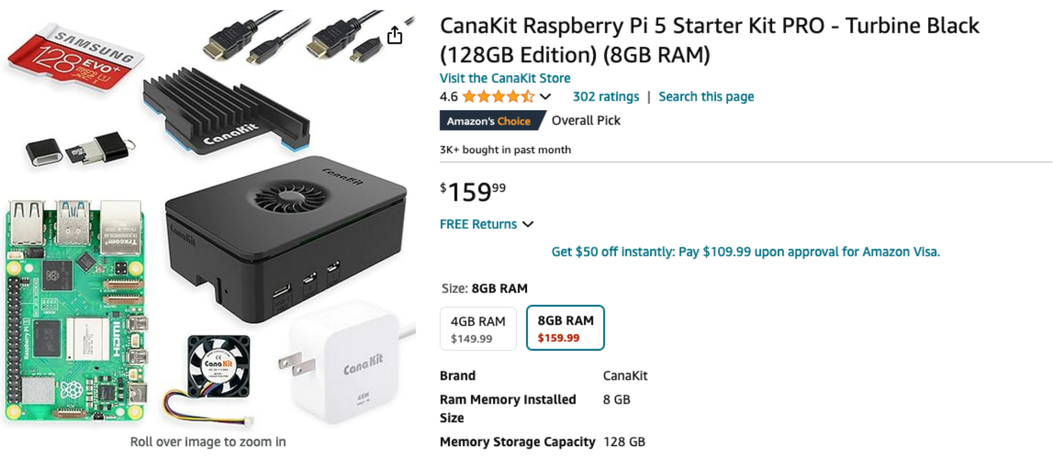
$239.00

FREE Returns

Get $50 off instantly: Pay $189.00 upon approval for Amazon Visa.

Color: **NUC11ATKC4 8GB+256GB**

Brand	waitloan
Operating System	Windows 11 Pro
CPU Model	Celeron
CPU Speed	2.9 GHz
Graphics Card	Integrated



CanaKit Raspberry Pi 5 Starter Kit PRO - Turbine Black (128GB Edition) (8GB RAM)

Visit the CanaKit Store

4.6 ★★★★☆ 302 ratings | Search this page

Amazon's Choice Overall Pick

3K+ bought in past month

$159.99

FREE Returns

Get $50 off instantly: Pay $109.99 upon approval for Amazon Visa.

Size: **8GB RAM**

Brand	CanaKit
Ram Memory Installed Size	8 GB
Memory Storage Capacity	128 GB

Expertise not required

- With ChatGPT anyone can learn to write code with Chat GPT
- You do not need to be a software engineer – although it helps

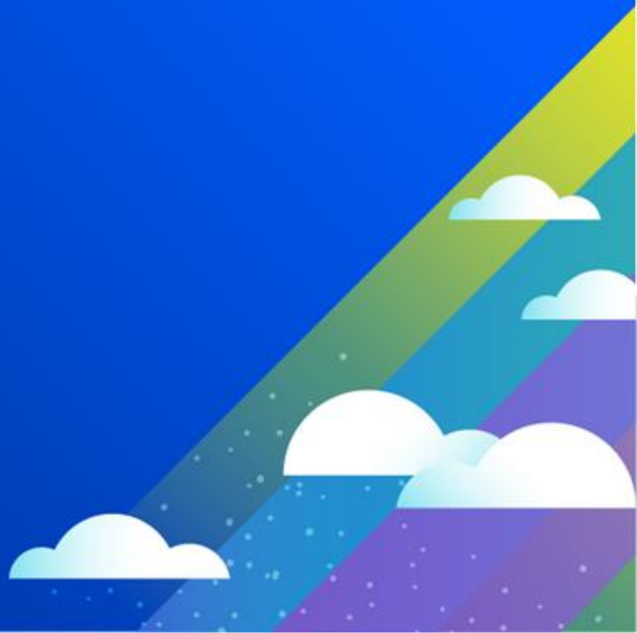
Lets get Started!

- Login to Existing ChatGPT Account
- If this is your first time using ChatGPT, we recommend that you attend the Introduction to ChatGPT session

Software Languages

ChatGPT will write in just about any software language.

Lets give it a try. In the prompt write “Using basic write a program that prints "Hello World" onto the screen”



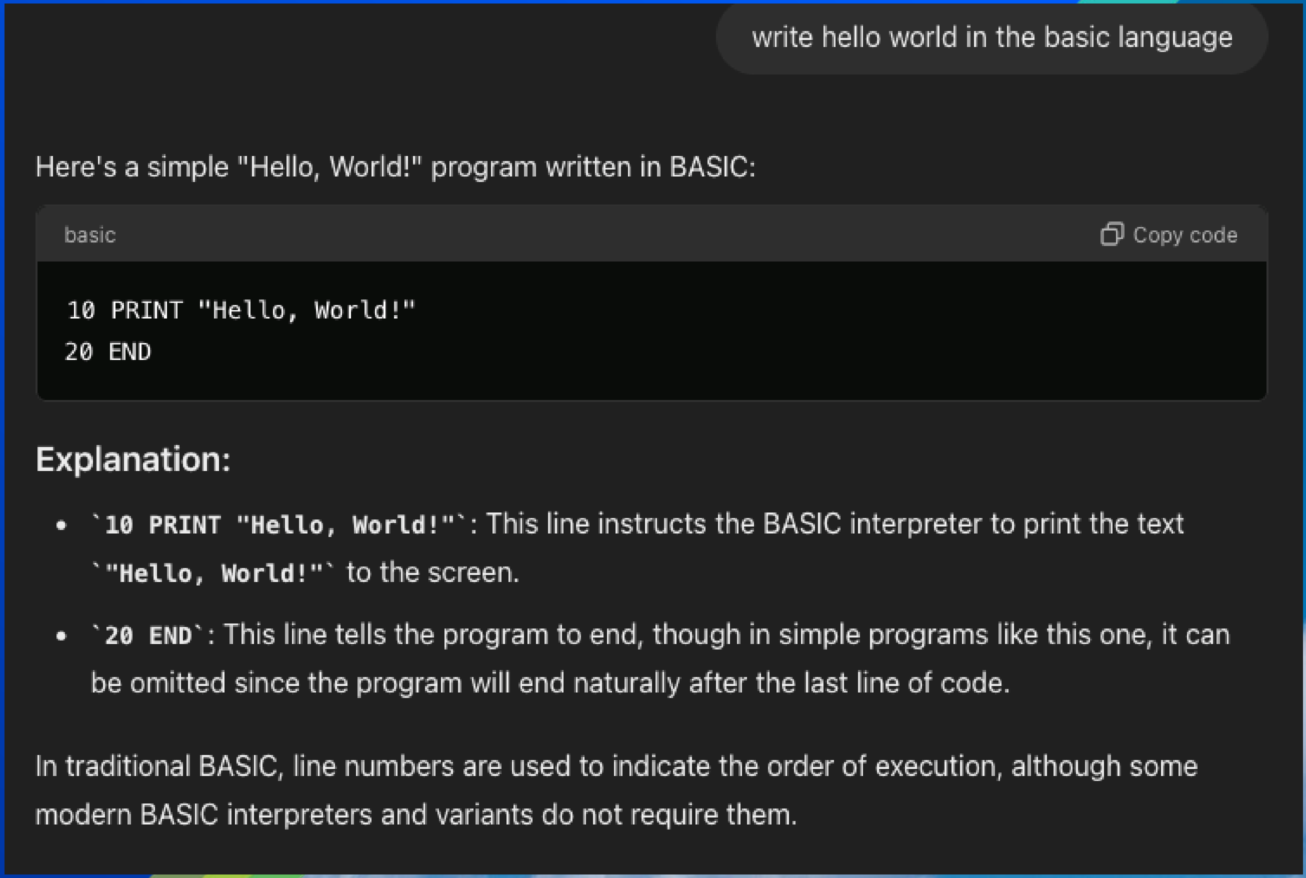
vmware EXPLORE

Basic – Hello World

- Lets take a moment and review the ChatGPT IO

The request to ChatGPT is called the Prompt.

- You will get some explanation of what the results will be
- You will get a Code Block that has a Copy Code button in the upper right Corner that can be used to copy and paste the results
- You will get an explanation of the above code



write hello world in the basic language

Here's a simple "Hello, World!" program written in BASIC:

```basic
10 PRINT "Hello, World!"
20 END
```

Explanation:

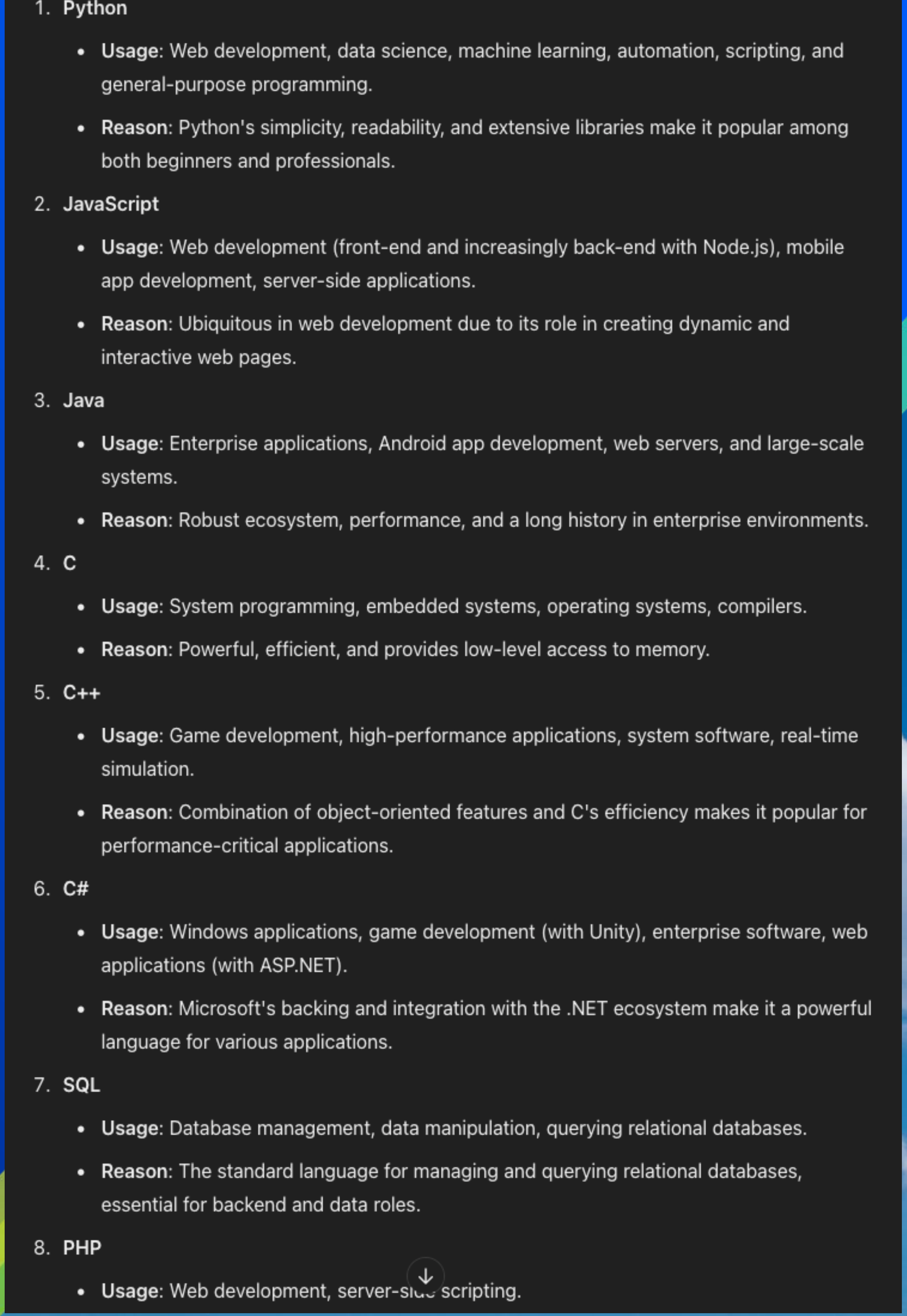
- `10 PRINT "Hello, World!"`: This line instructs the BASIC interpreter to print the text `"Hello, World!"` to the screen.
- `20 END`: This line tells the program to end, though in simple programs like this one, it can be omitted since the program will end naturally after the last line of code.

In traditional BASIC, line numbers are used to indicate the order of execution, although some modern BASIC interpreters and variants do not require them.

vmware EXPLORE

Software Languages

In the Room, what are the common languages that you write in?



1. **Python**
   - **Usage**: Web development, data science, machine learning, automation, scripting, and general-purpose programming.
   - **Reason**: Python's simplicity, readability, and extensive libraries make it popular among both beginners and professionals.
2. **JavaScript**
   - **Usage**: Web development (front-end and increasingly back-end with Node.js), mobile app development, server-side applications.
   - **Reason**: Ubiquitous in web development due to its role in creating dynamic and interactive web pages.
3. **Java**
   - **Usage**: Enterprise applications, Android app development, web servers, and large-scale systems.
   - **Reason**: Robust ecosystem, performance, and a long history in enterprise environments.
4. **C**
   - **Usage**: System programming, embedded systems, operating systems, compilers.
   - **Reason**: Powerful, efficient, and provides low-level access to memory.
5. **C++**
   - **Usage**: Game development, high-performance applications, system software, real-time simulation.
   - **Reason**: Combination of object-oriented features and C's efficiency makes it popular for performance-critical applications.
6. **C#**
   - **Usage**: Windows applications, game development (with Unity), enterprise software, web applications (with ASP.NET).
   - **Reason**: Microsoft's backing and integration with the .NET ecosystem make it a powerful language for various applications.
7. **SQL**
   - **Usage**: Database management, data manipulation, querying relational databases.
   - **Reason**: The standard language for managing and querying relational databases, essential for backend and data roles.
8. **PHP**
   - **Usage**: Web development, server-side scripting.

Lab #1

Have the AI write Hello world in each of these Languages

- Python
- Javascript
- Java
- C
- C++
- C#
- TypeScript
- Go (Golang)

- Don’t forget that you can request the AI to convert one language to another. For example “rewrite this code in [Java]”

User Rolls and Modifications

- When having the AI help you write code, you should be specific as to what Roll you want the AI to write in. Further, ask the AI to add other modifications

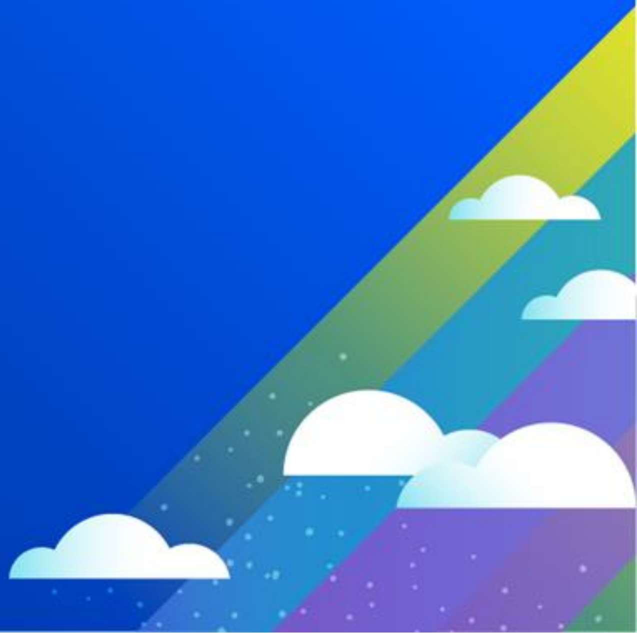
- Write the code as a beginner or as an expert.
- Follow best coding practices for the language of choice
- Explain how the code works
- Document the Code

Lab #2.

- Ask the AI to rewrite the Hello World with each and then all of these modifications:

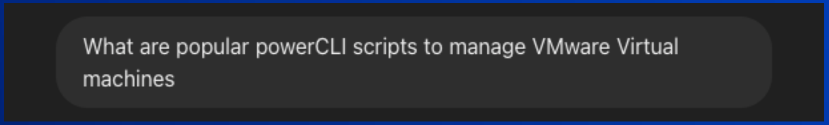
  Write the code as a beginner or as an expert.
- Follow best coding practices for the language of choice
- Explain how the code works
- Document the Code

How did the code change



vmware EXPLORE

ChatGPT – PowerCLi Intro


What are popular powerCLI scripts to manage VMware Virtual machines

```
1.   Get-VM | Select-Object Name, PowerState, NumCpu, MemoryMB, VMHost, @{N="IP
     Address";E={@($_.guest.ipaddress)}}
2.   Get-VM -Name "VM1","VM2","VM3" | Start-VM
3.   $sourceVM = Get-VM -Name "SourceVM"
     $targetVMName = "CloneVM”
     $targetDatastore = Get-Datastore -Name "Datastore1"
     $targetFolder = Get-Folder -Name "TargetFolder"
     $targetHost = Get-VMHost -Name "ESXiHost1" New-VM -Name $targetVMName -VM
     $sourceVM -VMHost $targetHost -Datastore $targetDatastore -Location $targetFolder
4.   Get-VM | Select Name, @{N="Tools Status";E={$_.ExtensionData.Guest.ToolsStatus}}
5.   $vm = Get-VM -Name "VMName"
     $adapter = Get-NetworkAdapter -VM $vm
     Set-NetworkAdapter -NetworkAdapter $adapter -NetworkName "NewNetwork" -Confirm:$false
6.   Get-VM -Name "VMName" | New-Snapshot -Name "SnapshotName" -Description "Snapshot before upgrade”
7.   Get-VM | Select Name,
     @{N="CPU Usage";E={(Get-Stat -Entity $_ -Stat cpu.usage.average -Realtime -MaxSamples 1).Value}},
     @{N="Memory Usage";E={(Get-Stat -Entity $_ -Stat mem.usage.average -Realtime -MaxSamples 1).Value}}
8.   Get-VM | Select Name,
     @{N="Provisioned Space (GB)";E={[math]::round($_.ProvisionedSpaceGB,2)}},
     @{N="Used Space (GB)";E={[math]::round($_.UsedSpaceGB,2)}}
9.   $vm = Get-VM -Name "VMName"
     $targetHost = Get-VMHost -Name "TargetHost"
     Move-VM -VM $vm -Destination $targetHost
10. $vm = Get-VM -Name "NewVM"
     $spec = New-OSCustomizationSpec -Name "CustomSpec" -Domain "domain.local" -ChangeSid
    Set-VM -VM $vm -OSCustomizationSpec $spec
```

The answer will be typical:

1. VM Inventory Report
2. Power On/Off Multiple VMs
3. Clone a VM
4. VMware Tools Status Check
5. Change VM Network Adapter
6. Snapshot Management
7. VM Performance Metrics Report
8. VM Disk Space Usage Report
9. Migrate VM to Another Host
10. VM Guest OS Customization

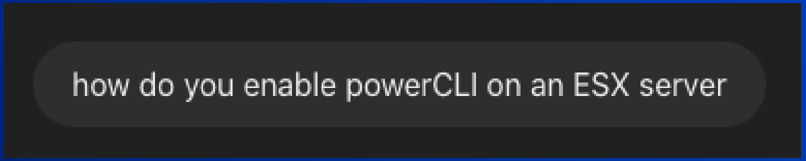
This does not however, make you a programmer ! …. Yet…

Lab #3.

Install PowerCLI and Connect to ESX

vmware® EXPLORE

ChatGPT – PowerCLi Intro


how do you enable powerCLI on an ESX server

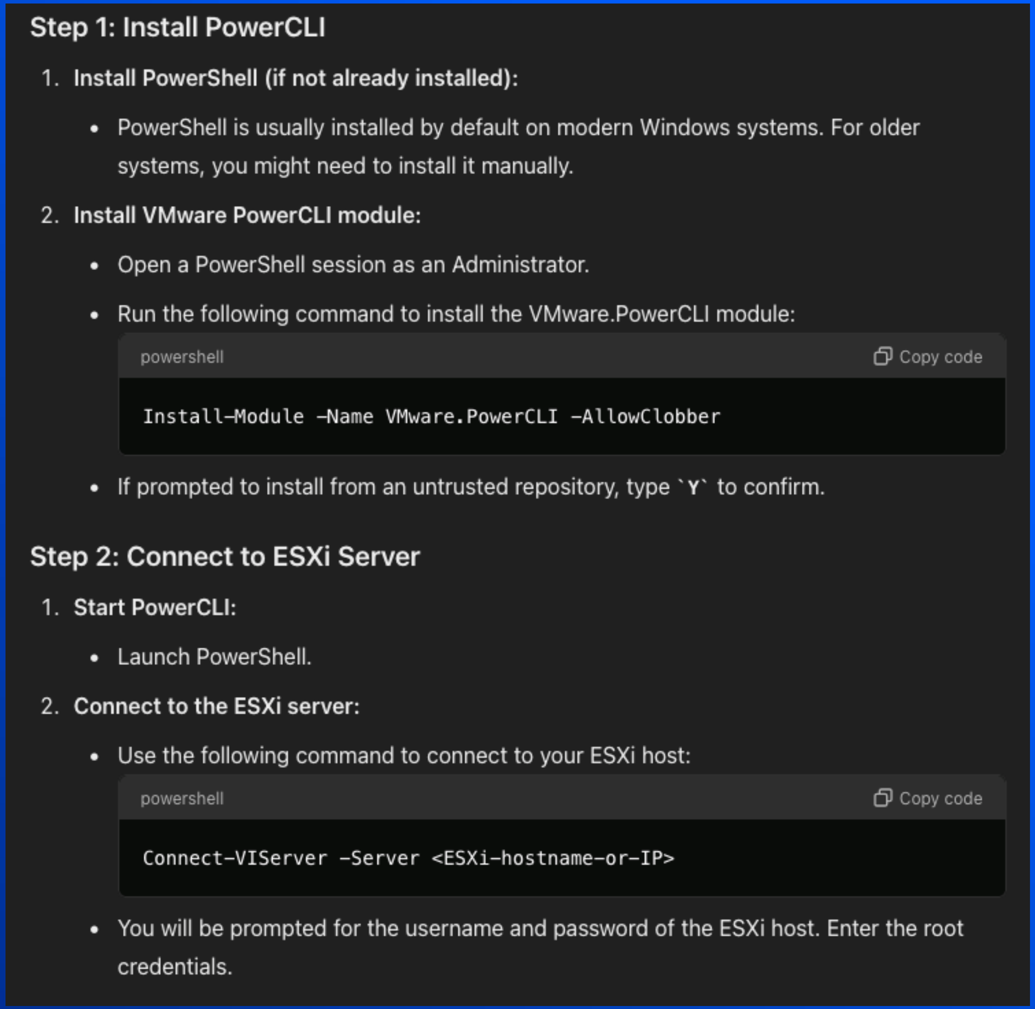
It's always good to understand how to enable your environment.

Simply ask ChatGPT

Notes:

- ✓ You must run PowerShell as Admin
- (Search Powershell and right click and "Run as admin" to launch Powershell as Admin)
- ✓ You must Enable Powershell Security

```
> Import-Module VMware.VimAutomation.Core
> Set-ExecutionPolicy –ExecutionPolicy RemoteSigned
> Set-PowerCLIConfiguration –InvalidCertificateaction Ignore –Confirm:$false
```


Step 1: Install PowerCLI

1. **Install PowerShell (if not already installed):**
   - PowerShell is usually installed by default on modern Windows systems. For older systems, you might need to install it manually.
2. **Install VMware PowerCLI module:**
   - Open a PowerShell session as an Administrator.
   - Run the following command to install the VMware.PowerCLI module:

```powershell
Install-Module -Name VMware.PowerCLI -AllowClobber
```

   - If prompted to install from an untrusted repository, type `Y` to confirm.

Step 2: Connect to ESXi Server

1. **Start PowerCLI:**
   - Launch PowerShell.
2. **Connect to the ESXi server:**
   - Use the following command to connect to your ESXi host:

```powershell
Connect-VIServer -Server <ESXi-hostname-or-IP>
```

   - You will be prompted for the username and password of the ESXi host. Enter the root credentials.

```
➢ Connect-VIServer –Server 72.172.185.171

➢ Name                              Port    User
➢ ---------                         -----   ------
➢ 72.172.185.171                    443     root
```

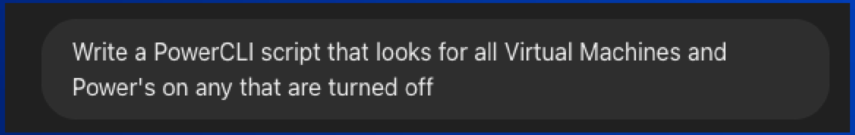
Simple PowerCLI Results

```
> Connect-VIServer –Server 72.172.185.171

Name                          Port       User
---------                    -------     -------
72.172.185.171                 443        root


> Get-VMHost

➢ Name                ConnectionState   PowerState    NumCpu  CpuUsageMhz CpuTotalMhz MemoryUsageGB MemoryTotalGB Version
➢ ---------          ----------------------  ----------------   -------------  -------------------  -----------------  ----------------------  ---------------------  ----------
➢   72.172.185.171    Connected         PoweredOn          2           275        3798             7.625           7.999    8.02

> Get-VM

➢ Name               PowerState  Num CPUs  MemoryGB
➢ ---------          ----------------  ---------------  ----------------
➢ CodeLabTable1      Powered On  1            2.000
➢ CodeLabTable2      Powered On  2            2.000
➢ CodeLabTable3      Powered On  3            2.000
➢ CodeLabTable4      Powered On  4            2.000
```

Lab #3.

Write a Basic PowerShell Program

ChatGPT – PowerCLi Intro



Write a PowerCLI script that looks for all Virtual Machines and Power's on any that are turned off

```
# Connect to vCenter Server
$vcenterServer = "your_vcenter_server" # Replace with your vCenter Server
$credential = Get-Credential
Connect-VIServer -Server $vcenterServer -Credential $credential

# Get all VMs
$vms = Get-VM

# Filter VMs that are powered off
$poweredOffVMs = $vms | Where-Object { $_.PowerState -eq "PoweredOff" }

# Power on each VM that is powered off
foreach ($vm in $poweredOffVMs) {
    Write-Host "Powering on VM: $($vm.Name)"
    Start-VM -VM $vm
}

# Disconnect from vCenter Server
Disconnect-VIServer -Confirm:$false
```

This is looking more like an App:

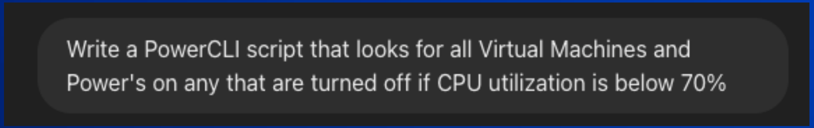
Programing is looking now looking more and more like knowing how to use ChatGPT

But,…. can we make this more complex..

See Lab3.ps1 for the source saved in your directory

vmware EXPLORE

ChatGPT – PowerCLi Intro


Write a PowerCLI script that looks for all Virtual Machines and Power's on any that are turned off if CPU utilization is below 70%

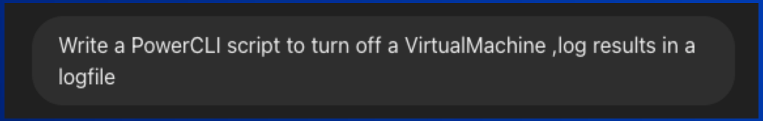
```
# Connect to the vCenter server
Connect-VIServer -Server "YourVCenterServer"

# Set the CPU threshold percentage
$cpuThreshold = 70

# Get all hosts and their CPU usage
$hosts = Get-VMHost | Select Name, @{Name="CpuUsage";Expression={($_.CpuUsageMhz / $_.CpuTotalMhz) * 100}}

# Iterate over each host
foreach ($host in $hosts) {
   # Check if the host's CPU usage is below the threshold
    if ($host.CpuUsage -lt $cpuThreshold) {
        Write-Host "Host $($host.Name) is under the CPU threshold with $($host.CpuUsage)% CPU usage."

        # Get all VMs on this host that are powered off
        $vms = Get-VM -Location $host.Name | Where-Object {$_.PowerState -eq "PoweredOff"}

        # Power on each VM that is powered off
         foreach ($vm in $vms) {
             Write-Host "Powering on VM: $($vm.Name)"
             Start-VM -VM $vm
          }
    } else {
        Write-Host "Host $($host.Name) is over the CPU threshold with $($host.CpuUsage)% CPU usage.
        No VMs will be powered on.”
    }
 }
# Disconnect from the vCenter server

Disconnect-VIServer -Confirm:$false
```

Can you make the next program:

1. Error checking ?
2. Time of Day ?
3. Results Table
4. Email transaction?
5. Log results

vmware EXPLORE

ChatGPT – PowerCLi Intro


Write a PowerCLI script to turn off a VirtualMachine ,log results in a logfile

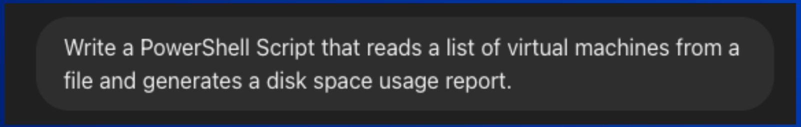
Can you make the next program:

1. Let’s make it log things.
2. Note the level of complixity with try/catch constructs.

```
# Set variables
$vCenterServer = "your_vcenter_server"
$vmName = "your_vm_name"
$logFile = "C:\path\to\logfile.txt"


# Function to log messages
function Log-Message {
   param (
      [string]$message
   )
   $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
   $logEntry = "$timestamp - $message"
   Add-Content -Path $logFile -Value $logEntry
}


# Connect to vCenter Server
try {
   Connect-VIServer -Server $vCenterServer -WarningAction Stop
   Log-Message "Connected to vCenter Server $vCenterServer."
} catch {
   Log-Message "Failed to connect to vCenter Server $vCenterServer. Error: $_"
   exit
}


# Get the virtual machine
$vm = Get-VM -Name $vmName -ErrorAction SilentlyContinue


if ($vm -eq $null) {
   Log-Message "Virtual machine '$vmName' not found."
   Disconnect-VIServer -Confirm:$false
   exit
}


# Power off the virtual machine
try {
   Stop-VM -VM $vm -Confirm:$false
   Log-Message "Virtual machine '$vmName' was powered off successfully."
} catch {
   Log-Message "Failed to power off virtual machine '$vmName'. Error: $_"
}


# Disconnect from vCenter Server
Disconnect-VIServer -Confirm:$false
Log-Message "Disconnected from vCenter Server $vCenterServer."
```

vmware EXPLORE

ChatGPT – PowerCLi Intro



Write a PowerShell Script that reads a list of virtual machines from a file and generates a disk space usage report.

Can you make the next program:

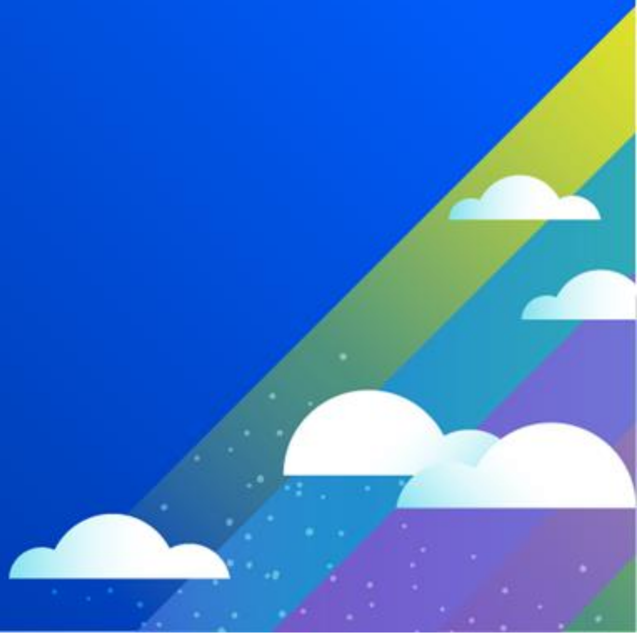
1. Lets read a list of VM’s from a file
2. Place results in a .csv file

```
# Variables
$vCenterServer = "your_vcenter_server"
$vmListFile = "C:\path\to\vm_list.txt"   # File containing VM names, one per line
$reportFile = "C:\path\to\DiskUsageReport.csv"  # Path to the report file


# Connect to vCenter Server
Connect-VIServer -Server $vCenterServer


# Initialize an array to hold the report data
$report = @()


# Read the VM names from the file
$vmNames = Get-Content -Path $vmListFile


# Loop through each VM
foreach ($vmName in $vmNames) {
  # Get the virtual machine object
  $vm = Get-VM -Name $vmName -ErrorAction SilentlyContinue


  if ($vm -eq $null) {
    Write-Host "Virtual machine '$vmName' not found." -ForegroundColor Red
    continue
  }

  # Get the disk space usage for each hard disk
  $disks = Get-HardDisk -VM $vm
  foreach ($disk in $disks) {
    # Calculate the disk space usage
    $capacityGB = [math]::Round($disk.CapacityKB / 1MB, 2)
    $usedSpaceGB = [math]::Round($disk.FreespaceGB - ($disk.CapacityGB - $disk.FreespaceGB), 2)
    $freeSpaceGB = [math]::Round($disk.FreespaceGB, 2)
    $usedPercentage = [math]::Round((($capacityGB - $freeSpaceGB) / $capacityGB) * 100, 2)


    # Create a custom object to store the disk usage data
    $reportItem = [PSCustomObject]@{
      VMName       = $vmName
      DiskName     = $disk.Name
      CapacityGB    = $capacityGB
      UsedSpaceGB   = $usedSpaceGB
      FreeSpaceGB   = $freeSpaceGB
      UsedPercentage = $usedPercentage
    }

    # Add the data to the report array
    $report += $reportItem
  }
}
$report | Export-Csv -Path $reportFile -NoTypeInformation
```

vmware EXPLORE

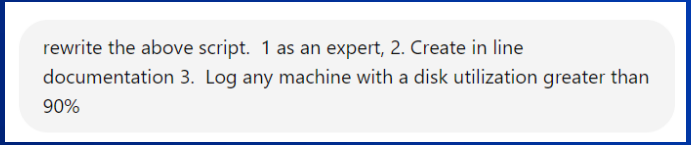
Lab #4.

Write a Complicated PowerShell Program



vmware EXPLORE

ChatGPT – PowerCLi Intro


rewrite the above script. 1 as an expert, 2. Create in line documentation 3. Log any machine with a disk utilization greater than 90%

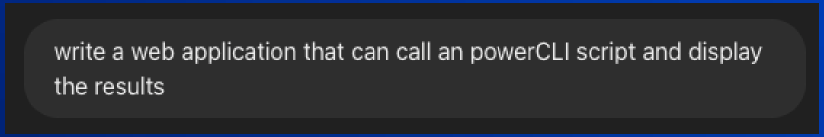
Ask the AI to rewrite the previous Lab and alter it to:
1 write it with as an expert
2 include documentation to better understand the code
3.Create a log of all the machines that disk are more than 90% full

```
# Connect to the vCenter server
$vCenterServer = "your_vcenter_server"
Connect-VIServer -Server $vCenterServer

# Define the path for the log file
$logFile = "C:\VM_Disk_Utilization_Log.txt"

# Clear the log file if it exists
if (Test-Path $logFile) {
    Remove-Item $logFile
}

# Get all VMs in the vCenter
$allVMs = Get-VM

# Iterate over each virtual machine
foreach ($vm in $allVMs) {
    # Check if the VM is powered on
    if ($vm.PowerState -eq "PoweredOn") {
        # Power off the VM
        Write-Host "Powering off VM: $($vm.Name)"
        Stop-VM -VM $vm -Confirm:$false
    }

    # Ensure the VM is powered off before proceeding
    while ($vm.PowerState -ne "PoweredOff") {
        Start-Sleep -Seconds 5
        $vm = Get-VM -Name $vm.Name
    }
```

Lab #5.

Write a Web Application that calls PowerCLI

ChatGPT – Multi Language

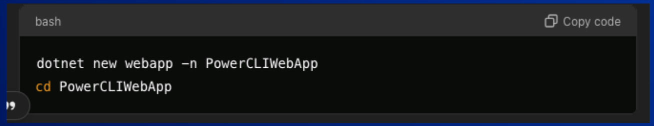


write a web application that can call an powerCLI script and display the results

1. Set Up the ASP.NET Core Web Application

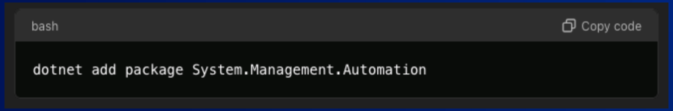
First, create an ASP.NET Core web application:

- **Create a new project**:
  1. Open Visual Studio or use the .NET CLI.
  2. Use the following command to create a new ASP.NET Core web application:

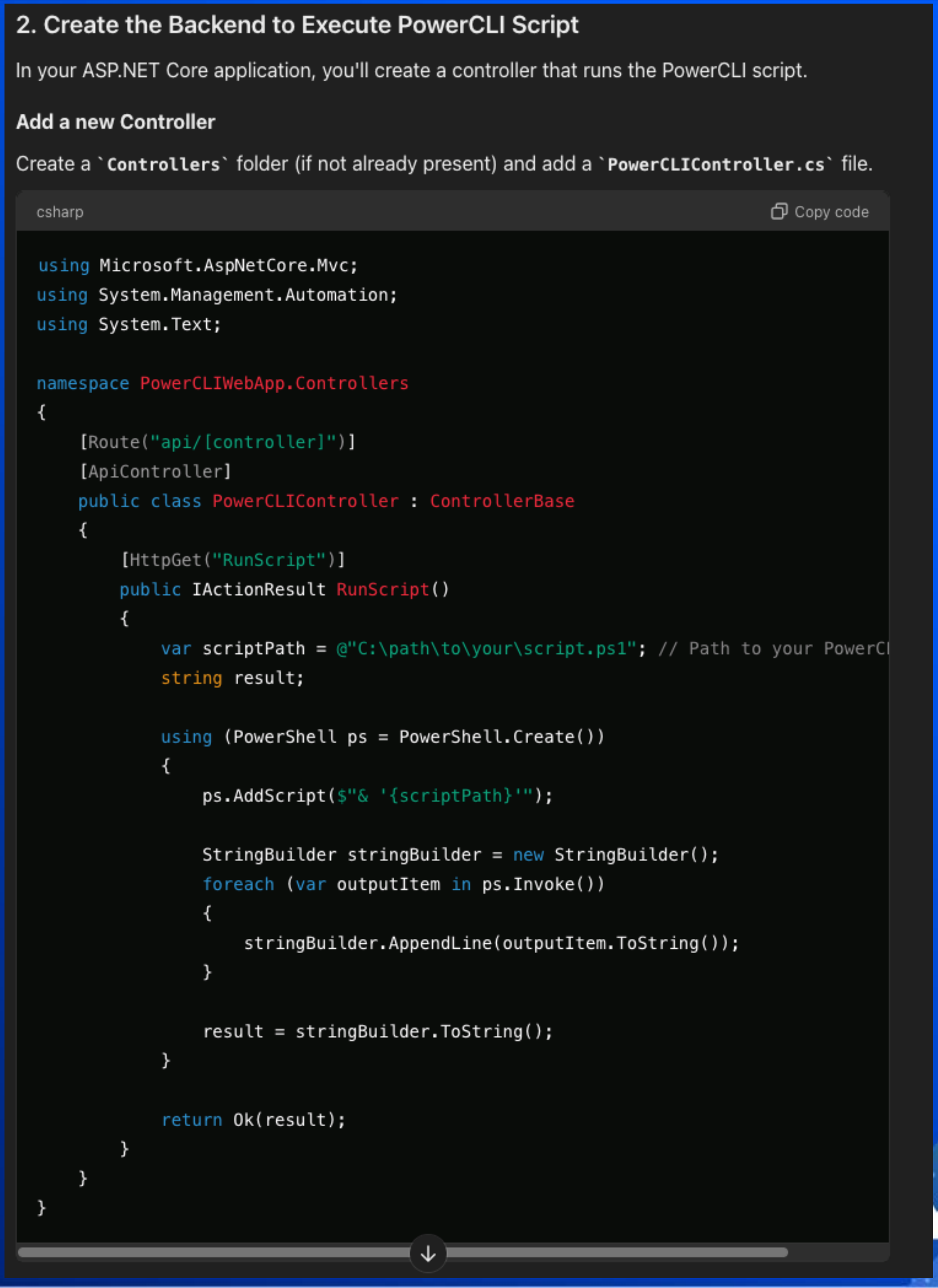


```bash
dotnet new webapp -n PowerCLIWebApp
cd PowerCLIWebApp
```

- **Install the required NuGet packages**:
  1. If using PowerShell from .NET, you need the System.Management.Automation package:



```bash
dotnet add package System.Management.Automation
```



2. Create the Backend to Execute PowerCLI Script

In your ASP.NET Core application, you'll create a controller that runs the PowerCLI script.

Add a new Controller

Create a `Controllers` folder (if not already present) and add a `PowerCLIController.cs` file.

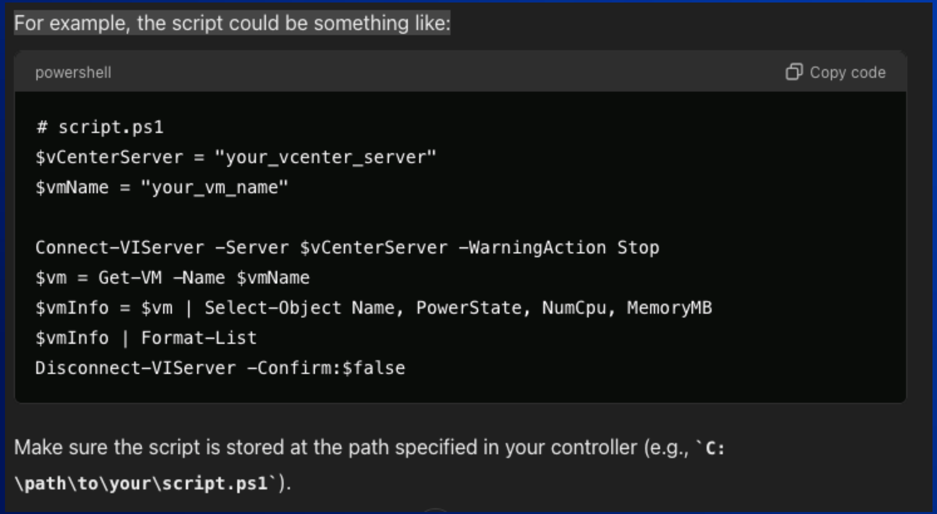
```csharp
using Microsoft.AspNetCore.Mvc;
using System.Management.Automation;
using System.Text;

namespace PowerCLIWebApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PowerCLIController : ControllerBase
    {
        [HttpGet("RunScript")]
        public IActionResult RunScript()
        {
            var scriptPath = @"C:\path\to\your\script.ps1"; // Path to your PowerC
            string result;

            using (PowerShell ps = PowerShell.Create())
            {
                ps.AddScript($"& '{scriptPath}'");

                StringBuilder stringBuilder = new StringBuilder();
                foreach (var outputItem in ps.Invoke())
                {
                    stringBuilder.AppendLine(outputItem.ToString());
                }

                result = stringBuilder.ToString();
            }

            return Ok(result);
        }
    }
}
```

vmware® EXPLORE

ChatGPT – Multi Language

3. Create the PowerCLI Script

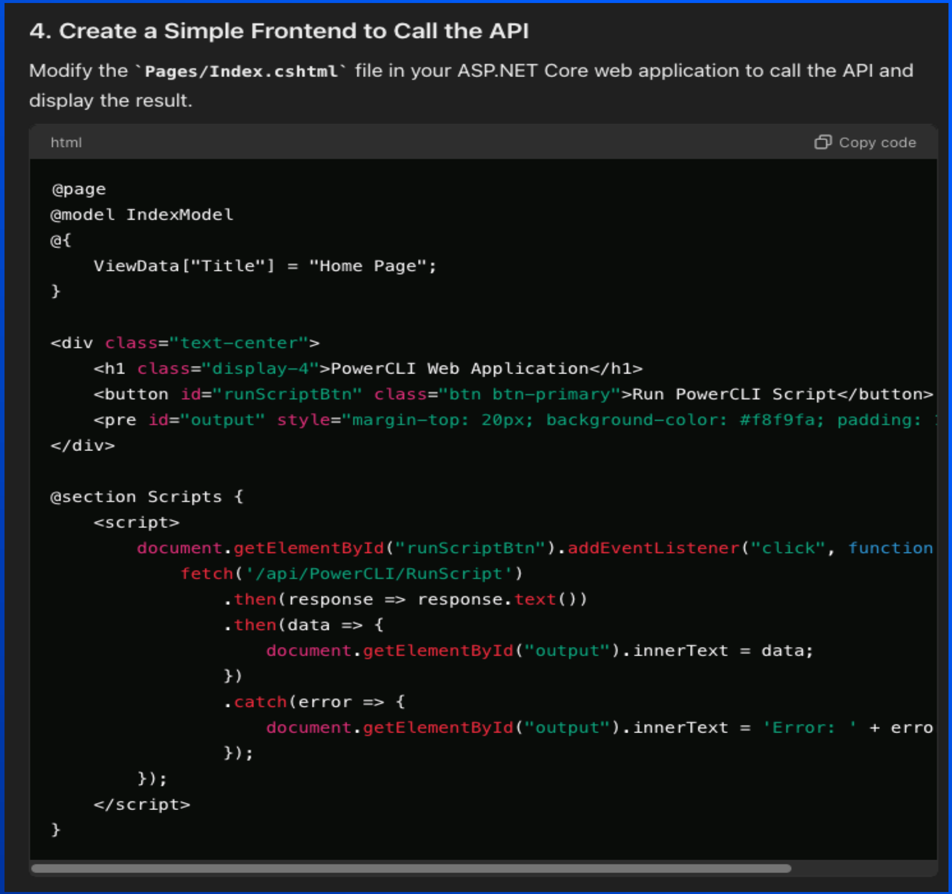
For example, the script could be something like:



For example, the script could be something like:

```powershell
# script.ps1
$vCenterServer = "your_vcenter_server"
$vmName = "your_vm_name"

Connect-VIServer -Server $vCenterServer -WarningAction Stop
$vm = Get-VM -Name $vmName
$vmInfo = $vm | Select-Object Name, PowerState, NumCpu, MemoryMB
$vmInfo | Format-List
Disconnect-VIServer -Confirm:$false
```

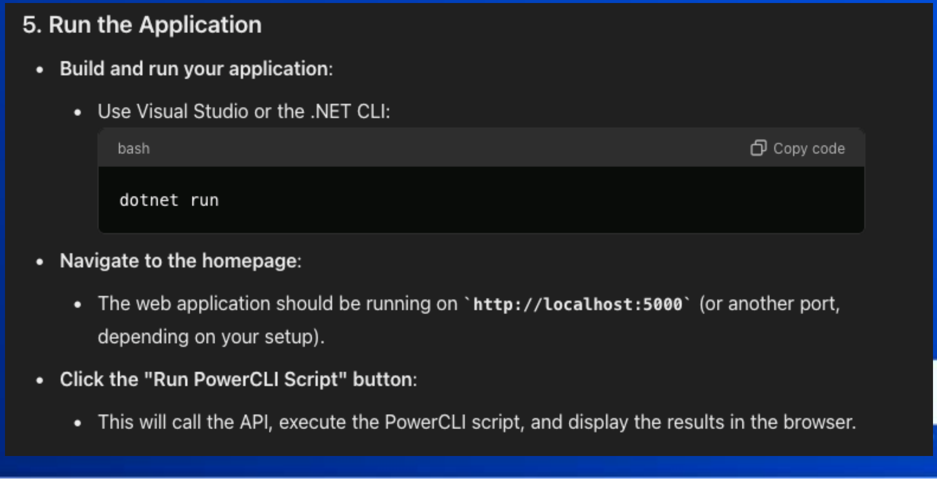
Make sure the script is stored at the path specified in your controller (e.g., `` `C:\path\to\your\script.ps1` ``).



4. Create a Simple Frontend to Call the API

Modify the `` `Pages/Index.cshtml` `` file in your ASP.NET Core web application to call the API and display the result.

```html
@page
@model IndexModel
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">PowerCLI Web Application</h1>
    <button id="runScriptBtn" class="btn btn-primary">Run PowerCLI Script</button>
    <pre id="output" style="margin-top: 20px; background-color: #f8f9fa; padding:
</div>

@section Scripts {
    <script>
        document.getElementById("runScriptBtn").addEventListener("click", function
            fetch('/api/PowerCLI/RunScript')
                .then(response => response.text())
                .then(data => {
                    document.getElementById("output").innerText = data;
                })
                .catch(error => {
                    document.getElementById("output").innerText = 'Error: ' + erro
                });
        });
    </script>
}
```



5. Run the Application

- **Build and run your application:**
  - Use Visual Studio or the .NET CLI:

```bash
dotnet run
```

- **Navigate to the homepage:**
  - The web application should be running on `` `http://localhost:5000` `` (or another port, depending on your setup).
- **Click the "Run PowerCLI Script" button:**
  - This will call the API, execute the PowerCLI script, and display the results in the browser.

Example Ideas for Self Paced Fun

1. Write a PowerCLI Script that allocates share limits for Virtual Machines
2. Write a PowerCLI Script that snapshots a VM once week for 52 weeks
   - Write a PowerCLI to create a snapshot of a Virtual Machine
   - Write a PowerCLI to remove a snapshot of a Virtual Machine
3. Write a PowerCLI Script that shows all a VMs extended properities
4. Write a PowerCLI Example to set a custom field for a VM
5. Write a PowerCLI Script to vMotion from one ESX server to another if CPU first ESX server CPU usage is over 80% and second ESX is below 50%, sleep for 1 hour then check again.

Bonus: Raspberry Pi and LiveCopper ModelC BMP 280 Temperature Sensors

1. Write a program in python that reads temperature from a BMP280 on a Raspberry Pi
2. Write a program in C that used the FTDI I2C library to get the temperature from a BMP280 sensor for Windows, the FTDI chip FT232h is being used.

Prompt Contest

Submit the most useful prompt to an powershell App:

eric.nielsen@Broadcom.com

Be sure to include your first and last name for the wall of fame.

Wall of Fame

Top Submissions!